Tuning Glovo's Dispatching Engine at Scale via Optimal Treatment Rules

David Masip and Ponç Palau

July 12, 2024

Tuning Glovo's dispatching engine at scale via optimal treatment rules

David Masip*
david.masip@glovoapp.com
Glovo
Barcelona, Spain

Ponç Palau**
ponc.puigdevall@glovoapp.com
Glovo
Barcelona, Spain

ABSTRACT

Glovo operates a three-sided marketplace connecting vendors, customers, and riders to facilitate order delivery. Central to this operation is the dispatching engine, which optimises real-time assignment of orders to riders. The dispatching engine employs a matching cost function to balance customer delivery times and rider efficiency. The matching cost function depends on a number of tunable coefficients that give more or less weight to several important operational metrics. This paper details Glovo's methodology for optimising the coefficients of the matching cost function, focusing on the optimisation pipeline used to determine optimal coefficient configurations and to test them in the real world. We use event-based simulation and multi-objective optimisation to tune these coefficients, ensuring a desirable trade-off between customer experience and operational efficiency. Additionally, we run switchback experiments and leverage optimal treatment rules to make informed roll-out decisions for new configurations, improving KPIs across diverse markets. Using real data from a Glovo experiment we showcase how our approach based on estimating optimal treatment rules demonstrates significant improvements over naive global roll-out strategies. Moreover, we report the results of a simulation study comparing the results of estimating the optimal treatment rules using several estimators available in the literature.

CCS CONCEPTS

• Mathematics of computing → Hypothesis testing and confidence interval computation; **• Computing methodologies →** *Genetic algorithms*; *Discrete-event simulation*; *Supervised learning by classification.*

KEYWORDS

Simulation, Optimisation, Genetic Algorithms, AB Testing, Switchback Testing, Optimal Treatment Rules

1 INTRODUCTION

Customers use Glovo's mobile or web app to place orders from vendors, that are then delivered by riders to the customer's location. Vendors include restaurants, grocery shops and also general retailers. With a footprint in 25 countries, hundreds of millions of orders are served every year in our platform.

At the core of Glovo's operation lies the dispatching engine, the system charged with assigning orders to riders in real time in each city we operate in. The dispatching engine: (i) solves a series of combinatorial optimisation problems to first generate possible sequences of orders to be delivered by a same rider, which we call routes, (ii) solves an assigment problem to assign routes to riders. This final stage which assigns routes to riders solves an assignment problem using a taylor-made objective function, which we call the matching cost function.

The matching cost function takes as inputs delivery time and distance travelled estimations from machine learning models. These estimations are weighted according to some tunable coefficents and then summed to form the final matching cost function. The choice of the matching cost function coefficients, which can be done individually for each city Glovo operates in, heavily influences operations, with trade-offs between different metrics arising naturally. In particular, prioritizing the lower customer delivery times can lead to inefficient use of the available rider fleet. On the other hand, optimising for fleet efficiency might increase customer delivery times, risking the delivery of cold food, and as a result, worsening customer experience.

In the present paper, we will deep-dive into our pipeline for optimising the coefficients in the matching cost function, with a particular emphasis on how we do experimentation to decide when and where to roll out changes. We will review how optimal roll out decisions are related to the estimation of optimal treatment rules ([13]) and in turn how the later are related to estimators for the conditional average treatment effect. Using both simulated and real data sets, we will assess the impact of the decisions made by optimal treatment rules versus more naive approaches, like a global roll-out decision.

The rest of the paper is organized as follows. In section 2 we briefly review related work by other players in our industry. In section 3 we provide a high level overview of a stylised version of the matching step in our dispatching engine. In section 4 we review how event based simulation can be leveraged to approximately tune the coefficients in the matching cost function. In section 5 we describe how we run and analyse experiments to decide in which geogaphies to roll out the new coefficients, showing how optimal treatment rules can be used to make these decisions, both in simulated and real data sets.

2 RELATED WORK

Using simulation to improve the operations in a platform like Glovo is a common practice in the industry. Some examples of this are the works of [3], where they present how to use simulation in order to explore possible improvements to the dispatching algorithms, specially in cases of undersupply. Following a similar fashion, in [2] they present how event-based simulation is used to test new features and improve current ones in their marketplace. While these articles focus on explaining the details of their simulation frameworks, our paper focuses on:

*Both authors contributed equally to this research.

- How to use simulation to optimise the dispatching engine of a global marketplace.
- How to select different configurations of the dispatching engine from a Pareto front based on regional inputs.

Our work uses the methodology in [9], [12] and [13] in an industry application. In particular, we use a slight modification of their framework where the unit of analysis, the unit of randomisation and the unit of rollout are all different. This is a common scenario in logistics applications, where the unit of randomisation is the switch, the unit of analysis is the order and the unit of rollout is the city.

3 OPTIMISING THE DISPATCHING ENGINE

To simplify the presentation, and to be able to focus on the optimisation of the coefficients in the matching cost function, we will present a highly simplified and stylised version of how rider-order matching works at Glovo. In particular, we will consider only the case in which individual orders (and not more complex routes) need to be assigned to riders, and a heavily simplified matching cost function.

Recall that the whole dispatching system, including the matching step, operates at a city level, and in near real time. Thus, in what follows we fix a given city, and a given moment during a day.

We let R denote the set of currently available riders in the city. Similarly, we let O be the set of all orders that currently need to be assigned. Let c_{ro} be the cost of assigning order o to the rider r and $x_{ro} \in \chi$ be our decision variable, taking the value of one if the rider r is assigned to order o and zero otherwise. We define an extra decision variable y_o indicating if the order o has been assigned to any rider. For every order-rider eligible matching (A), a cost c_{ro} is computed. For every order o, we define f_o representing the cost of not assigning it. Typically, f_o is considerably bigger than c_{ro}. Then, the following minimisation problem is solved to derive the matches:

$$\min_{x \in \chi} \sum_{(r,o) \in A} c_{ro} x_{ro} + \sum_{o \in O} (1 - y_o) \cdot f_o$$

s.t.

$$\sum_{r \in R} x_{ro} = y_o \quad \forall o \in O$$

$$\sum_{o \in O} x_{ro} \leq 1 \quad \forall r \in R$$

Constraints ensure that each order is assigned to at most one rider, and that each rider can be assigned at most one order. Not all the rider-order pairs are eligible. There are a number of constraints that are checked before a pair is considered eligible. For example, the distance between the pickup location and the position of the rider, vehicle constraints according to pickup and delivery areas, etc.

As stated in the introduction, the matching cost c_{ro} depends on multiple factors. To simplify the presentation, throughout the paper we will work with a simplified version, that depends on only three inputs:

- riderDistance$_{ro}$ Estimated distance traveled by the rider.
- riderDT$_{ro}$ Estimated time the rider r will spend delivering the order o. This term is used for controlling the quality of our service at peak times, when the number of orders can be considerable higher than the number of riders available.
- customerDT$_{ro}$ Estimated delivery time of order o if it is assigned to rider r, that is, the elapsed time since the order is created until the order is delivered.

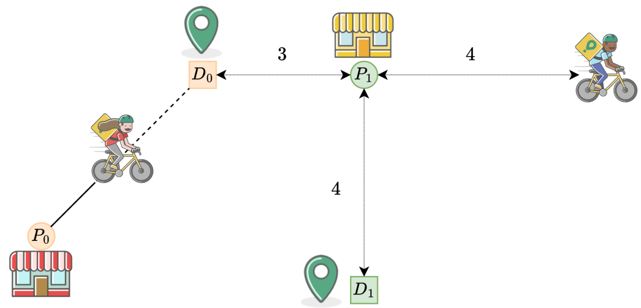

Figure 1: Scenario of two orders and two riders. Rider red (r_0) is delivering o_0, rider blue is idle waiting for orders to be assigned. Numbers indicate space units. Courier speed is 1 space unit/1 time unit.

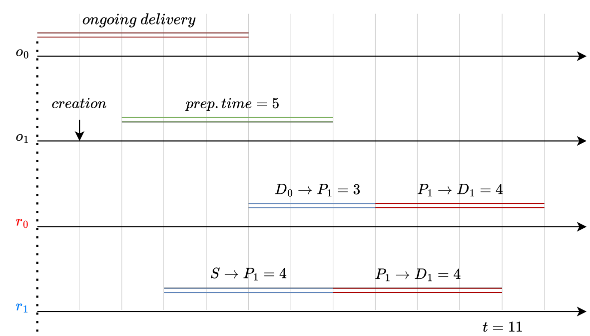

Figure 2: Order and rider timelines. The top two rows represent the timelines with order events. The bottom two rows represent the timelines with rider events. $X \leftarrow Y$ stands for the time that it takes for the rider to go from location X to location Y. S stands for starting position of the rider.

All three of these inputs are the predictions of in-house machine learning models. We define the matching cost c_{ro} as:

$$c_{ro} = \alpha_0 \cdot \text{riderDistance}_{ro} + \alpha_1 \cdot \text{riderDT}_{ro} + \alpha_2 \cdot \text{customerDT}_{ro}$$

To exemplify the importance of the values of the coefficients $\Omega = [\alpha_0, \alpha_1, \alpha_2]$, which weigh each of the terms in the cost c_{ro}, we present an illustrative scenario. Consider the setting depicted in Fig. 1, where we have two orders and two riders, namely rider red (r_0) and rider blue (r_1). At the current time of assignment, order 0 is already being delivered by the rider in red, while order 1 is still pending to be assigned.

Drawing a timeline of the different elements involved can provide a clearer visual representation of the various terms in the cost function, as shown in Fig. 2.

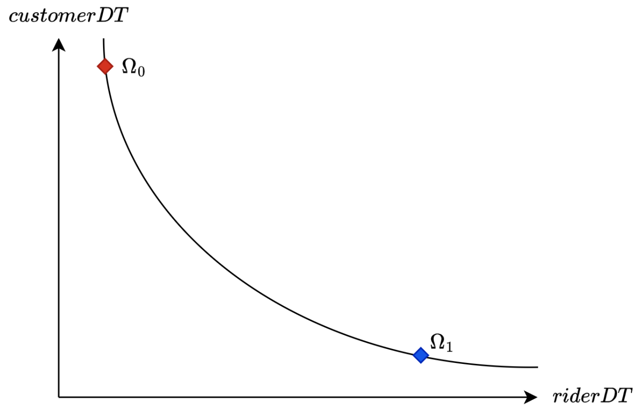

Figure 3: Pareto front highlighting the trade-off between rider and customer delivery time.

Based on the cost definition and the values provided in the example, the various terms of the cost function take on the following values:

- Pair $r_0 - o_1$
  - riderDistance$_{01}$ = 7
  - riderDT$_{01}$ = 7
  - customerDT$_{01}$ = 12

  Leading to $c_{01}(\Omega) = \alpha_0 \cdot 7 + \alpha_1 \cdot 7 + \alpha_2 \cdot 12$
- Pair $r_1 - o_1$
  - riderDistance$_{11}$ = 8
  - riderDT$_{11}$ = 8
  - customerDT$_{11}$ = 11

  Leading to $c_{11}(\Omega) = \alpha_0 \cdot 8 + \alpha_1 \cdot 8 + \alpha_2 \cdot 11$

Let us see how the assignments change as a function of Ω. Set $\Omega_0 = [1, 1, 1]$ and $\Omega_1 = [0.5, 0.5, 2]$. Then, for Ω_0 we have that $c_{01}(\Omega_0) = 26$ and $c_{11}(\Omega_0) = 27$ so we would select the red rider (r_0) in this case. On the contrary, for Ω_1 we have that $c_{01}(\Omega_1) = 31$ and $c_{11}(\Omega_1) = 30$, so we would select the blue rider (r_1).

If we look at different metrics for both cases, we see that when we use Ω_0, we are selecting the red rider, which delivers the order later than the blue rider, providing a worse experience for the final user, but in a more efficient way because it takes less time for the rider to deliver it. As commented before, a configuration of Ω that prioritises riderDistance and riderDT is extremely useful when there are significantly more orders than riders. Instead, if we assign orders using Ω_1, we are prioritising customer experience because we select the rider that can deliver the order the earliest.

Clearly, there is a trade-off between decreasing the time it takes for a rider to deliver an order (riderDT), or what is the same, delaying the dispatch because we are waiting for a rider that is better positioned to deliver that order more efficiently, and the delivery time of the order itself (customerDT). Therefore, as we change values of Ω, we have different results of riderDT and customerDT forming a Pareto front, such as the one depicted in Fig. 3.

In general, the point in the Pareto front we want to operate at for a given market is a strategic business decision. Changing the coefficents in the matching cost function is one way to guarantee we (approximately) operate at the targeted trade-off between rider and customer delivery time. In order to determine which combination of coefficients yields a desired trade-off, we leverage our in-house logistics simulator, which we describe next.

4 SIMULATION-BASED OPTIMISATION

Our simulator allows us to approximately evaluate the effect of different configurations of Ω in the KPIs of interest without the need of running expensive experiments.

The simulation is performed inside the logistics simulator, a tool designed and implemented internally in Glovo. This simulator is able to approximately recreate the operations of a city given some historical data of it and a matching cost function configuration Ω. Given that in other occasions we already published details about our dispatching-simulator, in this article we only provide a high level explanation. For more details, please see [1].

For a given city and date, the simulator loads the following historical data:

- Riders:
  - Check-in/Check-out times to simulate fleet availability
  - Starting locations in the city for each rider
- Orders: Historical activation time, pickup and delivery locations.

Next, given an input configuration of Ω it produces assignments for the orders. Please note that once we change the values of Ω, we cannot trust the location of the riders in the historical data because the dispatching engine could have assigned different orders to them, changing the final location of the riders at the end of one delivery. Finally, the simulator provides a report with the average KPIs of interest for that entire day, such as the average delivery time of the orders, the average time riders spend delivering the orders, among others. In what follows, we will refer to the KPIs obtained from a simulation as avgCustomerDT for the average delivery time of all the orders in the simulated date and avgRiderDT for the average time it took riders to deliver all the orders in the simulated date. Our simulator can either be run for a single date or for a range of dates, providing the average KPIs for all the dates in the range.

Within this context, we use the simulator as a black box system that provides us with the estimated values of city level KPIs of interest for a given configuration of Ω for a set of dates. More precisely, we designed a pipeline that runs a multi-objective optimisation algorithm to find the Pareto fronts of different Ω. The algorithm in use is the NSGA-II ([6]) algorithm, which is a well-known algorithm for multi-objective optimisation problems.

The complete process of the pipeline is depicted in Fig. 4. In order to prevent overfitting of the coefficients of the cost function to the historical data of choice, we use two sets of dates: training and testing dates. We run the NSGA-II algorithm and obtain Pareto fronts for all the cities in the training dates. Then, we simulate with every Ω configuration in the latter Pareto front the behaviour of the cities during the testing dates, and we obtain a set of metrics of interest averaged over these dates.

We are now interested in selecting a single configuration from the Pareto front for each city. In the following section, we present our methodology to do so.

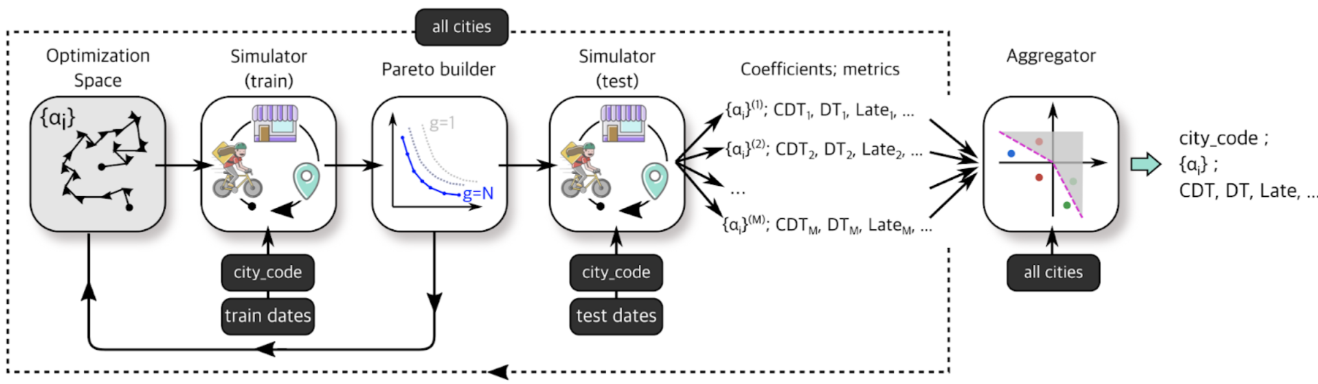

Figure 4: Entire pipeline for finding the optimal configuration Ω for a city.

4.1 Selecting configurations from the Pareto front

In order to introduce the methodology to choose the configuration Ω, we first introduce some notation. For every city $c \in C$ we have a set of configurations $S_c = \{\Omega_1, \Omega_2, \ldots, \Omega_i, \ldots, \Omega_n\}$ and a set of metrics $\{\text{avgCustomerDT}_i, \text{avgRiderDT}_i\}$ for each Ω_i.

We assume that cities in the same region have a shared set of business objectives, and therefore, the final configuration for every city in the region must satisfy the same constraints. Regions are defined as a set of countries; a particular case of it is a single country. Usually, since we have a trade-off between different logistic metrics, we are interested in minimising one of them while not compromising the others. For example, we could be interested in minimising the average customer delivery time while not compromising the average rider delivery time. In what follows, without loss of generality, we assume that we are indeed interested in minimising customer delivery time with a constraint on how much we can increase rider delivery time.

In our methodology, we first define a set of constraints that the average simulated rider delivery time of that region must satisfy. This ensures, that, on average over the whole region, rider delivery time is not too deviated from the business objective. We then solve an optimisation problem that minimises customer delivery time while respecting the constraint on the average rider delivery time. Before formally defining the optimisation problem, let us recall some variables and define some new ones:

- C: set of cities in the region, indexed by c.
- S_c: set of configurations of Ω for city c, indexed by i.
- $\Delta\text{riderDT}_{ci}$: difference between simulated rider delivery time for city c and Ω_i and the simulated rider delivery time for city c and the configuration that is in production for that city.
- $\Delta\text{customerDT}_{ci}$: difference between simulated rider delivery time for city c and Ω_i and the simulated rider delivery time for city c and the configuration that is in production for that city. In this case, the metric we are trying to minimise.
- f_c: Fraction of orders in the region that are created in city c.
- x_{ci} : binary variable that indicates if we select Ω_i for city c.
- L: maximum deviation allowed for the counter-metric, in this example, riderDT.

We then define the following minimisation problem:

$$\begin{aligned}
\min_{x_{ci}} \quad & \sum_{c \in C} \sum_{i \in S_c} x_{ci} \cdot \Delta\text{customerDT}_{ci} \cdot f_c \\
\text{s.t.} \quad & \\
& \sum_{i \in S_c} x_{ci} = 1 \quad \forall c \in C, \\
& \sum_{c \in C} \sum_{i \in S_c} x_{ci} \cdot \Delta\text{riderDT}_{ci} \cdot f_c \leq L,
\end{aligned}$$

Since we know the simulator is not perfect, in order to decide whether we should roll out the newly optimised matching cost function coefficients, we run randomised experiments. In the next section, we describe how this experiments are set up and analysed.

5 EXPERIMENT DESIGN AND ROLL-OUT

The matching problem is solved at the city level, which implies that exposure to changes in the matching algorithm, such as changes in the matching cost function coefficients, can only be randomised at the city level. Because of this, we run switchback experiments ([4]) where, for each day and city, we randomly allocate the city to treatment (new coefficients) or control (old coefficients). The goal of these experiments is to evaluate whether the change in the coefficients will result in a positive impact to a given target metric Y, which we assume is defined at the order level. Without loss of generality, we assume that positive changes in Y are positive for our business.

The switchback experiment design allows us to estimate the average treatment effect on the target metric Y in multiple ways, for example, using the difference in means estimator, and at multiple levels of aggregation: for each city individually, per region, globally, etc. Leveraging these estimates we need to make decisions on where to rollout the optimised coefficients.

5.1 Global roll-out decision

One way to decide to roll-out the new coefficients is checking if the global average treatment effect Δ is positive. This leads us to pose the following hypothesis testing problem:

- Null Hypothesis (H_0): $\Delta \leq 0$
- Alternative Hypothesis (H_1): $\Delta > 0$

Using this test, the new configurations are rolled out in all cities if we reject the null hypothesis. Clearly, this decision may be sub-optimal. It is possible that in some subset of cities the average effect is positive while in the rest of the cities it is negative. In this case, the optimal decision would be roll-out only in cities where the expected effect is positive, whereas the global roll-out decision will either be to roll-out in all the cities or in none. This is an instance of the known issue of treatment effect heterogeneity.

However, there are some scenarios where only a global roll-out decision makes sense from a product perspective. For instance, when testing a new user onboarding flow, we would, at some point, need to have all the new users consuming the same onboarding flow, in order to avoid maintaining different pieces of software. For the matching cost function optimisation problem, we do not have this issue, since we can have different configurations for different cities, and this does not increase the maintainance complexity at all.

Being able to make changes to the matching cost function at the regional or city level also makes change management a lot easier. Representatives from regional teams, owning the operations over a given region, will in general not accept to roll out a feature that can jeopardise the operations in their cities just for the good of the global average. In the following section, we show how optimal treatment rules ([9], [12], [13]) can be used to decide in which cities we should roll-out the new coefficients.

5.2 Optimal treatment rules roll-out

After running a switchback experiment like described in the previous section, we have the following data, where each observation corresponds to a delivered order:

- c_i: The city corresponding to observation i.
- X_{c_i}: A set of features that describe the city c_i, and only vary at the city level.
- T_i: The treatment assignment on observation i.
- Y_i: The outcome on metric Y on observation i.

Let $\mathcal{X}$ denote the range of X_{c_i} and consider a function $d : \mathcal{X} \to \{0, 1\}$. Such a function d represents a possible roll-out or treatment rule, which, based on the city characteristics, tells us whether we should roll out the new coefficients or not. If $d(X_c) = 1$ then, according to rule d, we should roll-out in any city with features X_c and if $d(X_c) = 0$ we should not. The decision rule that encodes a global roll-out decision is the rule d that is always equal to 1. Let $Y(d)$ is the potential outcome in the world in which treatment allocation is decided according to the function d.

The optimal treatment rule is defined as the function d that maximises the expected value of the potential outcome $Y(d)$, that is

$$d^* = \arg \max_{d:\mathcal{X}\to\{0,1\}} E\left[Y(d)\right],$$

It is easy to show that $d^*(X_c) = 1$ if and only if $E\left[Y \mid T = 1, X_c\right] \geq E\left[Y \mid T = 0, X_c\right]$. Thus, estimating $d^*(X_c)$ is closely related to estimating the conditional average treatment effect (CATE), and any estimator of the CATE can be leveraged to estimate an optimal treatment rule.

Interestingly, the optimal treatment rule can also be estimated directly, without having to go through the estimation of the CATE. One way to do so that we have found very useful and easy to implement was proposed in [13]. Under the usual SUTVA [7] assumptions, for any $d : \mathcal{X} \to \{0, 1\}$,

$$E\left[Y(d)\right] = E\left[\frac{I_{d(X_c)=T} \cdot Y}{\pi_d(X)}\right],$$

where for any $x \in \mathcal{X}$, $\pi_d(x) = P(T = d(x))$. For designed experiments, which is our case of interest, the function π is known. Moreover, it is often constant and equal to 1/2. In this case, it follows that

$$\begin{aligned} d^* &= \arg \max_{d:\mathcal{X}\to\{0,1\}} E\left[I_{d(X_c)=T} \cdot Y\right] \\ &= \arg \min_{d:\mathcal{X}\to\{0,1\}} E\left[I_{d(X_c)\neq T} \cdot Y\right] \end{aligned}$$

Now,

$$E\left[I_{d(X_c)\neq T} \cdot Y\right]$$

can be thought of as a missclasification loss, where cases are weighted according to Y. A classifier can then be trained to minimise a smooth and/or convex proxy of this loss. More precisely, we can train a classifier with T as a target label, Y as a weight and X_c as features. This classifier is then an estimate of the optimal treatment rule d^*. In our case, we choose to use boosted trees [5] as the base learning algorithm. We then use the trained model to predict the treatment assignment for each city, and we roll-out the new coefficients in the cities where the model predicts a positive treatment assignment. We will refer to this estimator of the optimal treatment rule as the classifier based method, which is the one we use in practice.

5.3 Optimal treatment rules impact assessment

In order to assess the impact of the optimal treatment rules, the naive methodology is to take the subset of cities S where we decide to rollout and estimate the average treatment effect in there via:

$$\tilde{\Delta}_S = \frac{\sum_{i=1}^n Y_i \cdot T_i \cdot I_{i\in S}}{\sum_{i=1}^n T_i \cdot I_{i\in S}} - \frac{\sum_{i=1}^n Y_i \cdot (1 - T_i) \cdot I_{i\in S}}{\sum_{i=1}^n (1 - T_i) \cdot I_{i\in S}}$$

this is the average treatment effect in the cities where we rollout. We estimate the impact in all cities

$$\tilde{\Delta} = \tilde{\Delta}_S \cdot \frac{\sum_{i=1}^n I_{i\in S}}{n} \tag{1}$$

where n is the total number of orders and $\sum_{i=1}^n I_{i\in S}$ is the number of orders in the subset of cities S. This estimator of the impact is biased, as we are using the same data to decide in which cities we should rollout and to estimate the impact of the rollout. In order to correct this bias, we use a cross-fitting methodology([10], [11], [14]). We split the data in K folds, and in each split:

- We train the model in $K - 1$ folds and predict the treatment assignment in the K-th fold.
- We estimate the impact using the $\tilde{\Delta}$ in the K-th fold.

We then average the estimates from each fold to get an unbiased estimator of the impact of the optimal treatment rules rollout decision. A detail that is important to mention is that, in the cross-fitting methodology, we use grouped splitting at switch level, that is, data from every time period over which treatment allocation did not change in a given city remains in the same fold. If this is not done, the impact is going to be positively biased again, because of the intra-cluster correlation within switches, where treatment is constant.

In the next section we report the results of a simulation study we ran to compare the performance of multiple estimates of the optimal treatment rules. We will see that both with synthetic and real data, that decisions based on estimates of optimal treatment rules always outperform the global roll-out decision described in Section 5.1. One possible issue with our methodology is that we are targeting at a single metric, while in practice we have counter metrics that we want to keep at a certain level. In our experiments, we address this by combining all metrics in an overall evaluation criterion. To keep things simpler, the following sections assume that we have a single target metric, which is customer delivery time.

5.4 Simulations

5.4.1 Synthetic data simulation. We generate data using the following process:

- We use data from 30 Glovo cities, and we take as the target metric to optimise the customer delivery time.
- For each city, we take three features that describe the city, X_{c1}, X_{c2} and X_{c3}.
- We generate a city treatment effect $\tau_c \sim X_{c1} * X_{c2} + X_{c3} + U(-0.03, 0.05)$.
- We randomly assign treatment T_i to each order with a probability of 0.5.
- For treatment orders, we modify customer delivery time via $Y_i = Y_i \cdot (1 + \tau_c)$.

We will use this data to assess how different rollout rules perform in a controlled environment. We compare

- the classifier based approach of [13] to the estimate optimal treatment rulles,
- the method based on estimating the CATE to estimate optimal treatment rulles, using the X-learner and the T-Learner from [8],
- the global rollout rule, and
- the infeasible rule that performs a roll out in a city only when its real treatment effect τ_c is positive.

We refer to the later infeasible rule as the optimal rollout.

Let $\Delta^{optimal}$ be the impact of the optimal rollout. We can also compute the impact of the global rollout, Δ^{global}, by applying 1 in all the cities. For any treatment rule, like the classifier approach, the method based on T-Learner, or the method based on the X-Learner, we can compute the impact of the rollout, Δ^{rule}, by applying 1 in the cities where the treatment rule decides to roll-out.

We run 1000 realizations of the data generation process and we compute the average impact of the optimal rollout, global rollout, the classifier approach, the T-Learner, and the X-Learner. We also provide 95% confidence intervals for the impact of each rollout strategy, which are shown in Table 1.

Table 1: Cross-fitting scores for synthetic data

Rule	Mean Impact	2.5%	97.5%
Optimal	0.0131	0.0088	0.0181
XLearner	0.0130	0.0088	0.0180
TLearner	0.0130	0.0088	0.0180
Classifier	0.0127	0.0080	0.0176
Global	0.0009	-0.0080	0.0101

We can see that the global rollout is very far away, impact-wise, from the optimal rollout (more than 1% on average). Both the T-Learner and X-Learner are the closest to the optimal rollout. The mean effect difference between the T-Learner and the classification approach is of around 0.03%, which is very small compared to the global rollout. In the next section we see that, in a real experiment, the classification approach provides more stable cross-validation results.

5.4.2 Real data simulation. We now repeat a similar analysis with data from a matching cost function swithcback experiment run at Glovo. We use the following set of features to train the classification model:

- Pre-experiment aggregated metrics: Metrics that describe the city before the experiment, like average courier delivery time, average customer delivery time, average saturation, etc.
- City exogenous features: Features that describe the city, like population, country, size, etc.
- Control coefficients: The matching cost function coefficients of the control group.
- Treatment coefficients: The matching cost function coefficients of the treatment group.
- Changes in coefficients: The ratio between the treatment and control coefficients.

In this case we don't have a ground truth of the cities that should be rolled out, but we can compare the cross-validation impact of the optimal treatment rules with the global rollout impact. A global rollout decision would have increased our target metric by 0.6%, while the optimal treatment rules rollout would have increased it by 1.4%. Using the T-Learner would have decreased the target metric by 0.8%. For this reason, we prefer to use the classification approach, since it provides more stable cross-validation results. In the ideal scenario, we see that both of them are comparable, giving a slight edge to the CATE estimators, but in a scenario where we are missing many features explaining the CATE, the classification approach is more stable.

Table 2 shows the impact of the 5 splits in the cross-fitting routine.

We can also see how the scores produced by the optimal treatment rules classifier allow us to rank the cities by their treatment effect. Fig. 5 shows the average actual treatment effect in each bin. The ordering is not perfect, but we can see that the optimal

Table 2: Cross-fitting scores for real data

Classification test scores	T-Learner test scores
0.74%	-0.69%
0.93%	-1.43%
5.17%	2.05%
0.21%	-3.78%
0.34%	-0.20%

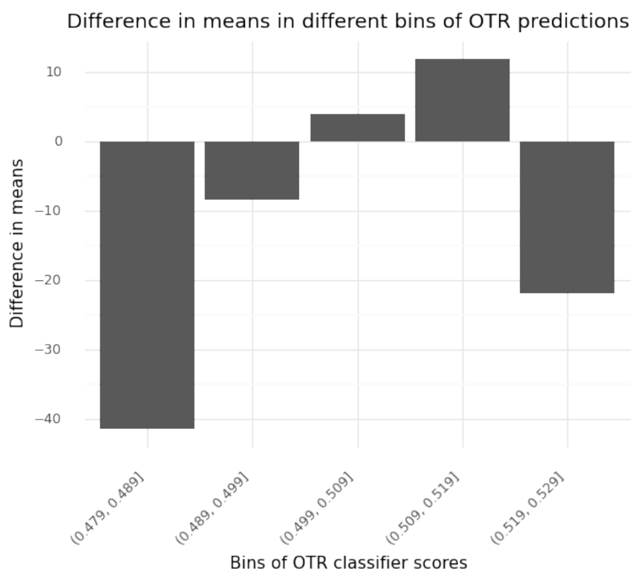

Figure 5: Difference in means by bins of optimal treatment rules classifier scores

treatment rules framework is still able to rank the cities by their treatment effect in most of the bins.

5.5 Discussion

In this paper we present the methodology we use to tune the matching cost function coefficients in a logistics context. We show how we use simulation-based optimisation to find a Pareto front of different configurations of the matching cost function coefficients. We then present a methodology to select a single configuration of the coefficients for each city, based on region level constraints. We also show how we run experiments to assess the impact of the new coefficients, and how we use optimal treatment rules to decide in which cities we should rollout the new coefficients.

One issue with our methodology is that, although we use optimal treatment rules for the analysis of the experiment, we do not design the experiment to have statistical power in a way that is coupled with the usage of optimal treatment rules. This is a flaw of our approach, where we may have under-powered experiments for the optimal treatment rules analysis. We are working on improving the design of the experiments to decide experiment length based on the optimal treatment rules analysis.

6 AUTHORS AND AFFILIATIONS

Ponç Palau is with Glovo, Barcelona, Spain.
David Masip is with Glovo, Barcelona, Spain.

ACKNOWLEDGMENTS

The authors would like to thank Glovo for providing the data and the resources to carry out this work. The authors would also like to thank Ezequiel Smucler, Alex Goldhoorn, Julia Barrufet, Francesco Ricci, Gonzalo Cordova, Manuel Bolivar, Thomas Cambier, Meghna, Alejandra Vendrell, Fernanda Rachel Del Bel Tavares, Able Johnson, Pedram Hajesmaeeli, Guillermo Izquierdo and Alejandro Tombazzi, for their contributions to this project and support.

REFERENCES

[1] [n. d.]. Glovo-simulator. https://medium.com/glovo-engineering/how-to-simulate-a-global-delivery-platform-7aa5fa475d88.
[2] [n. d.]. Simulated-marketplace. https://www.uber.com/en-ES/blog/simulated-marketplace/.
[3] [n. d.]. Simulating a Ridesharing Marketplace. https://medium.com/@adamgreenhall/simulating-a-ridesharing-marketplace-36007a8a31f2.
[4] Iavor Bojinov, David Simchi-Levi, and Jinglong Zhao. 2022. Design and Analysis of Switchback Experiments. arXiv:2009.00148 [stat.ME]
[5] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*. ACM. https://doi.org/10.1145/2939672.2939785
[6] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation* 6, 2 (2002), 182–197. https://doi.org/10.1109/4235.996017
[7] Guido W. Imbens and Donald B. Rubin. 2015. *Causal Inference for Statistics, Social, and Biomedical Sciences: An Introduction.* Cambridge University Press.
[8] Sören R. Künzel, Jasjeet S. Sekhon, Peter J. Bickel, and Bin Yu. 2019. Metalearners for estimating heterogeneous treatment effects using machine learning. *Proceedings of the National Academy of Sciences* 116, 10 (Feb. 2019), 4156–4165. https://doi.org/10.1073/pnas.1804597116
[9] Eric B Laber, Daniel J Lizotte, Min Qian, William E Pelham, and Susan A Murphy. 2014. Dynamic treatment regimes: technical challenges and applications. *Electronic Journal of Statistics* 8, 1 (2014), 1225–1272. https://doi.org/10.1214/14-ejs920
[10] Anton Schick. 1986. On Asymptotically Efficient Estimation in Semiparametric Models. *The Annals of Statistics* 14, 3 (1986), 1139 – 1151. https://doi.org/10.1214/aos/1176350055
[11] A.W. van der Vaart. 1998. *Asymptotic Statistics.* Cambridge University Press. https://books.google.es/books?id=fiX9ngEACAAJ
[12] Yingqi Zhao, Donglin Zeng, Eric B Laber, and Michael R Kosorok. 2015. New Statistical Learning Methods for Estimating Optimal Dynamic Treatment Regimes. *J. Amer. Statist. Assoc.* 110, 510 (2015), 583–598. https://doi.org/10.1080/01621459.2014.937488
[13] Y. Zhao, D. Zeng, A. J. Rush, and M. R. Kosorok. 2012. Estimating Individualized Treatment Rules Using Outcome Weighted Learning. *J. Amer. Statist. Assoc.* 107, 449 (Sep 2012), 1106–1118. https://doi.org/10.1080/01621459.2012.695674
[14] Wenjing Zheng and Mark J. van der Laan. 2010. Asymptotic Theory for Cross-validated Targeted Maximum Likelihood Estimation. 273 (Nov. 2010). https://biostats.bepress.com/ucbbiostat/paper273